Plan du Rapport sur les Stratégies IA pour le Jeu de Hex

1.Introduction

Présentation du jeu de Hex comme terrain

2.Compréhension du Projet et des Stratégies de Jeu

Aperçu des stratégies : RandomPlayer, MiniMax, MiniMaxPlus

(Alpha-Beta Pruning), MonteCarloPlayer, QLearningPlayer.

3.Évaluation et Analyse des Performances

Méthodologie : affrontements contre RandomPlayer sur un plateau de taille 8.

Présentation des résultats et analyse comparative des performances.

4. Réponses aux Questions

Ajout et justification des choix des heuristiques

Description de l'approche Monte-Carlo et sa différence avec les autres approches en terme de performence

5.Difficultés Rencontrées et Gestion des Complexités du Projet

Gestion de la complexité croissante avec l'intégration de nouvelles stratégies.

Importance de la gestion des versions pour le suivi des modifications et la collaboration.

6.Conclusion

Synthèse des apprentissages et des succès obtenus.

Réflexion sur l'importance de l'approche méthodique et adaptative dans les projets d'IA.

Lien vers le dépôt GitHub:

7. Mode d’emploi

1.Introduction

Le jeu de Hex offre un terrain d'expérimentation idéal pour l'application et le test de différentes stratégies d'intelligence artificielle. Dans ce projet, nous plongeons dans l'univers de Hex en utilisant le code fourni pour créer et tester des stratégies automatisées capables de jouer et, idéalement, de gagner contre des adversaires humains et artificiels.

Le projet se divise en trois parties essentielles : comprendre et démarrer avec une stratégie basique qui joue de manière aléatoire, introduire l'intelligence avec une stratégie MinMax et ses optimisations, et enfin, peaufiner les performances avec des heuristiques spécifiques et une recherche de Monte-Carlo en option pour les plus aventureux.

Ce rapport présente le processus de développement, les défis surmontés et les résultats obtenus, illustrant l'efficacité des stratégies conçues dans le contexte captivant du jeu de Hex.

2. Compréhension du Projet et des Stratégies de Jeu

Ce projet vise à explorer une gamme de tactiques pour le jeu de Hex à travers l'élaboration d'une série de stratégies distinctes, incarnées par la classe PlayerStrat et ses dérivées. Chaque sous-classe est conçue pour représenter une méthode de jeu particulière, fournissant une variété d'approches, des sélections aléatoires basiques à des calculs algorithmiques avancés. Voici une description améliorée des stratégies mises en œuvre :

Stratégie RandomPlayer : Fondamentale et intuitive, cette tactique sélectionne les coups au hasard. Elle constitue un étalon pour mesurer la performance relative des méthodes plus sophistiquées.

Stratégie MiniMax : Adoptant une perspective prévoyante, le MiniMax vise à optimiser le pire scénario favorable, en explorant les issues potentielles jusqu'à une certaine profondeur de l'arbre de décisions.

Stratégie MiniMaxPlus (Élagage Alpha-Beta) : Perfectionnement du MiniMax, cette technique réduit le volume de branches à considérer grâce à l'élagage Alpha-Beta, ce qui accroît l'efficience tout en préservant l'intégrité des choix tactiques.

Stratégie MonteCarloPlayer : Utilisant des simulations aléatoires pour évaluer les mouvements les plus opportuns, cette stratégie excelle dans les contextes où l'espace de possibilités est immense. Sa force réside dans son habileté à apprécier les configurations complexes sans avoir besoin d'un examen exhaustif.

Stratégie QLearningPlayer : Innovante, cette approche s'appuie sur l'apprentissage par renforcement, en affinant ses options stratégiques à la lumière des résultats antérieurs, ce qui lui permet d'affiner continuellement ses performances.

Ces méthodes constituent un écosystème diversifié de stratégies, chacune apportant sa propre perspective et ses avantages uniques dans le défi intellectuel qu'est le jeu de Hex.

3. Evaluation et Analyse des performences

Dans la phase expérimentale initiale, toutes les stratégies développées pour le projet d'IA Hex seront testées en compétition avec la stratégie RandomPlayer. Ces affrontements se tiendront sur un plateau de jeu de taille 8x8, avec une série de 10 parties pour chaque paire de stratégies confrontées. L'objectif principal est d'évaluer la performance de chaque stratégie face à une approche de jeu fondée sur le hasard.

	score	temps total des 6 parties
MiniMax	6/10	20 mn
MiniMaxPlus	8/10	10 mn
Monte-Carlo	8/10	8 mn
Qlearning	7/10	1 mn

Résultats du tournoi entre la stratégie Random et les autres stratégies pour 10 parties sur un plateau de taille 8

Les stratégies Minimax avec élagage Alpha-Beta (Minimaxab), Monte-Carlo et QLearning ont toutes montré une supériorité marquée par rapport à la stratégie de base Random, pas seulement en termes de taux de victoire mais également en efficience temporelle. Il est remarquable de constater que les stratégies Random et QLearning se distinguent par leur rapidité d’exécution. Le QLearning, en particulier, excelle par sa vitesse, ce qui est probablement dû à sa capacité à apprendre et à s'adapter de

manière accélérée. Ces observations suggèrent que les techniques heuristiques, d'élagage et d'apprentissage par renforcement représentent des voies prometteuses pour accroître les performances dans le jeu de Hex.

À présent, procédons à une comparaison entre Minimax et MinimaxPlus au cours de 10 parties sur un plateau de taille 8x8

	MinimMaxPlus	Minimax
score	9	1
pourcentage	9/10	1/10

Résultats du tournoi entre les stratégies MinimaxPlus et MiniMax sur 10 parties

D'après ces résultats, MiniMaxPlus a surpassé Minimax dans une série de simulations de jeu, remportant les 6 parties jouées, ce qui suggère un avantage significatif. Cet avantage est probablement dû aux différences dans leur mise en œuvre. La classe MiniMaxPlus intègre une fonction heuristique avancée, 'proximity_to_victory_heuristic', qui évalue à quel point le joueur est proche de remporter la partie dans Hex. Cette fonction accorde un score plus élevé aux configurations proches de la victoire, prenant en compte les chemins presque complets. En comparaison, la classe MiniMax utilise des évaluations plus basiques du plateau.

En plus, MiniMaxPlus applique l'élagage alpha-bêta à sa fonction minimax, optimisant ainsi la recherche de mouvements optimaux par la réduction du nombre de nœuds examinés dans l'arbre de jeu. Cette optimisation permet de parcourir l'arbre de décision plus efficacement, en identifiant potentiellement de meilleurs coups sans avoir à explorer l'ensemble des possibilités, particulièrement dans le cas d'une profondeur de recherche étendue.

La classe MiniMaxPlus tire également parti de sa fonction heuristique personnalisée et de la mesure de la proximité de la victoire pour mieux évaluer le plateau, ce qui peut lui conférer une capacité supérieure à anticiper des mouvements conduisant rapidement à un avantage stratégique.

En conclusion, la performance dominante de MiniMaxPlus indique que les améliorations apportées à son algorithme, y compris l'intégration de l'heuristique de proximité de victoire et l'utilisation de l'élagage alpha-bêta, lui octroient une supériorité tactique notable sur la version standard de Minimax.

Nous examinerons ces améliorations plus en détail dans la section 4, 'Réponses aux questions'.

Pour une analyse approfondie des performances, un tournoi round-robin sera organisé où chaque stratégie sera confrontée à toutes les autres à travers une série de 10 matchs, tous disputés sur un plateau de taille 10x10. Cette méthode nous fournira une comparaison directe des forces et des faiblesses de chaque stratégie dans divers contextes de jeu, offrant ainsi une évaluation exhaustive et précise de leur efficacité et adaptabilité relatives.

	MinimMaxPlus	Monte-Carlo
score	8	2
poucentage	8/10	2/10

Résultat du tournoi entre MiniMaxPlus et Monte-Carlo sur un jeu de taille 10

	MinimMaxPlus	Qlearning
score	7	3
total	7/10	3/10

Résultat du tournoi entre MiniMaxab et Qlearning sur un jeux de taille 10

	Monte-Carlo	Qlearning
score	6	4
poucentage	6/10	4/10

Résultats du tournoi entre Monte-Carlo et Qlearning sur un jeux de taille 26

Durant les essais intensifs mettant en compétition diverses stratégies d'intelligence artificielle dans des jeux de stratégie de tailles différentes, des tendances claires se sont dessinées en matière de performance et de réactivité.

La stratégie Random effectue des sélections de mouvements aléatoires et réagit extrêmement vite, indépendamment de la dimension du plateau de jeu. Sa vélocité est due à l'absence de nécessité de procéder à des analyses prédictives ou heuristiques, la rendant quasi-instantanée, bien qu'elle manque clairement de finesse stratégique.

La stratégie Q-Learning, quoique plus élaborée que Random, affiche aussi une grande rapidité de réaction. Apprenant de ses expériences, elle affine sa performance au fil des parties. Elle a prouvé sa capacité à prendre des décisions promptes, même sur des plateaux de jeu étendus, tels ceux de 26 cases par côté,

suggérant une généralisation efficace de l'apprentissage des configurations de jeu pour des décisions rapides sans analyses profondes à chaque coup.

La stratégie Monte-Carlo, qui repose sur des simulations aléatoires pour anticiper les issues des mouvements, est aussi relativement véloce. Bien qu'elle ne soit pas aussi instantanée que Random, elle surclasse souvent MiniMax en termes de délai de réponse, tout en offrant un jeu plus subtil et stratégique que la stratégie aléatoire.

Quant à MiniMaxPlus, une version améliorée de MiniMax, elle est reconnue pour sa capacité à identifier et contrer efficacement les mouvements potentiellement gagnants de l'adversaire. Toutefois, il a été observé que MiniMaxPlus tend à ralentir significativement avec l'accroissement de la profondeur de recherche, particulièrement au-delà de trois niveaux, et ce phénomène est d'autant plus marqué sur des plateaux de grande taille. Cela implique que, bien que MiniMaxPlus soit tactiquement supérieure grâce à son élagage alpha-bêta et ses heuristiques avancées, son utilité peut être restreinte par des contraintes temporelles dans des jeux de grande envergure ou nécessitant une réaction immédiate.

Ces constats mettent en lumière l'importance de l'équilibre entre rapidité de calcul et profondeur stratégique, surtout dans le cadre de jeux étendus ou exigeant des réponses en temps réel. Il est crucial de sélectionner la stratégie adéquate en fonction du contexte spécifique du jeu, en prenant en compte la complexité de la situation, le temps disponible pour la prise de décision et le rapport entre l'exactitude et la célérité requises

4.Réponses aux questions

Dans le cadre de l'amélioration de la méthode MiniMax pour le jeu de Hex, nous avons introduit des heuristiques spécifiques pour évaluer plus efficacement les positions sur le plateau. Ces heuristiques ont été conçues pour refléter les aspects stratégiques

cruciaux du jeu, permettant à l'algorithme de prendre des décisions plus informées.

Custom Heuristic : Cette heuristique est fondamentale pour Hex. Elle évalue le plateau en comptant le nombre de pièces alignées pour chaque joueur. L'accent est mis sur les lignes horizontales, verticales et diagonales où un joueur a placé ses pièces. Cette approche est logique dans Hex car l'objectif est de créer un chemin continu de pièces d'un bord à l'autre du plateau. Plus il y a de pièces alignées, plus on est proche de cet objectif.

Proximity to Victory Heuristic : Pour renforcer l'évaluation stratégique, j'ai intégré une heuristique avancée qui mesure la proximité d'un joueur à remporter la partie. Cette heuristique évalue les positions en fonction de leur potentiel à former un chemin gagnant. Elle attribue un score élevé aux positions qui sont proches de compléter un chemin gagnant et qui ne sont pas bloquées par l'adversaire. Cette approche est cruciale dans Hex, car elle permet d'évaluer non seulement la présence des pièces sur le plateau mais aussi leur utilité stratégique. Les chemins presque complets sont particulièrement importants, car ils augmentent significativement les chances de gagner.

Dans la méthode `evaluate_board`, ces deux heuristiques sont combinées pour évaluer les positions. Un score extrêmement élevé est attribué en cas de victoire imminente, tandis que les positions actuelles sont évaluées en fonction de la qualité et des perspectives stratégiques offertes. Cette combinaison offre un équilibre entre l'évaluation de l'état actuel du jeu et l'anticipation des

possibilités futures, ce qui est essentiel pour des décisions stratégiques éclairées dans le jeu de Hex.

En conclusion, ces heuristiques permettent d'évaluer les positions non seulement sur la base de critères quantitatifs, comme le

nombre de pièces, mais aussi en considérant des aspects qualitatifs tels que la formation stratégique de chemins et la proximité de la victoire. Cette approche plus nuancée confère à mon algorithme MiniMax une capacité de jugement améliorée, cruciale pour exceller dans un jeu complexe comme Hex.

La principale différence de performance entre Monte-Carlo et les autres stratégies comme Random et MiniMaxPlus est que Monte-Carlo peut évaluer plus efficacement les jeux de grande taille et les situations complexes. Contrairement à Random, qui ne prend pas en compte l'état du jeu, Monte-Carlo simule des fins de jeu pour obtenir une évaluation statistique de la meilleure action. En comparaison avec MiniMaxPlus, Monte-Carlo ne nécessite pas une recherche profonde dans l'arbre de jeu, ce qui le rend plus rapide, en particulier sur les plateaux de grande taille où la profondeur de recherche élevée de MiniMaxPlus ralentit considérablement les performances.

5.difficultés Rencontrées et Gestion des Complexités du Projet

Le développement du projet Hex IA a présenté des défis progressifs en termes de complexité. L'intégration de nouvelles stratégies, ainsi que l'affinement continu des classes existantes, a conduit à une complexité accrue dans la gestion des dépendances. Cette situation a parfois entravé notre rapidité d'exécution et exigé un niveau de vigilance accru pour préserver la cohérence du système.

La gestion des versions s'est avérée être un outil inestimable dans ce contexte, nous permettant de maintenir une organisation rigoureuse du code malgré la complexité grandissante. L'utilisation de plateformes telles que GitHub a été essentielle, non seulement

pour le suivi des modifications mais aussi pour la collaboration et la révision de code.

6.Conclusion

Le projet de développement de stratégies d'intelligence artificielle pour le jeu de Hex a été une aventure à la fois stimulante et enrichissante. Nous avons exploré un large éventail de stratégies, depuis les choix aléatoires simples de RandomPlayer jusqu'aux méthodes plus complexes et nuancées comme MiniMaxPlus et MonteCarloPlayer, en passant par l'approche innovante de QLearningPlayer. Chaque stratégie a révélé des aspects uniques du jeu de Hex et a offert des perspectives différentes sur la manière de relever ses défis stratégiques.

Nos expérimentations et analyses ont démontré que, si des stratégies simples comme RandomPlayer peuvent être efficaces en termes de vitesse, elles manquent de la profondeur stratégique nécessaire pour exceller dans Hex. En revanche, des approches plus sophistiquées comme MiniMaxPlus et MonteCarloPlayer ont montré une capacité supérieure à naviguer dans la complexité du jeu, offrant ainsi une meilleure performance globale. L'introduction d'heuristiques spécifiques et de l'élagage alpha-bêta a permis à MiniMaxPlus de surpasser la stratégie MiniMax standard, soulignant l'importance de l'optimisation dans la conception des algorithmes d'IA.

La réalisation de ce projet a également mis en évidence la nécessité d'une gestion efficace des ressources et de l'organisation, en particulier dans un contexte de complexité croissante. L'utilisation de la gestion des versions a été cruciale pour maintenir la cohérence du code et faciliter la collaboration. Le dépôt GitHub du projet est devenu un outil essentiel, non seulement

pour le suivi des progrès, mais aussi comme un moyen de partager nos découvertes et notre savoir-faire.

En somme, ce projet nous a permis de développer des compétences précieuses en matière de programmation, de résolution de problèmes et de pensée stratégique. Les leçons apprises ici sont largement applicables, dépassant le cadre du jeu de Hex pour toucher d'autres domaines de l'intelligence artificielle et de la stratégie de jeu. Nous sommes impatients de voir comment ces connaissances pourront être appliquées et développées dans de futurs projets.

Pour plus de détails sur le projet, les stratégies implémentées, et les résultats obtenus, n'hésitez pas à consulter notre dépôt GitHub :

git@github.com:khadimfall2/HexIAProject.git

https://github.com/khadimfall2/HexIAProject.git

7 Mode d'emploi

Pour lancer le projet de jeu de Hex avec des stratégies d'intelligence artificielle, suivez ces étapes simples :

1. **Activer l'environnement virtuel Python** Ouvrez un terminal et naviguez jusqu'au répertoire du projet sur votre bureau en utilisant la commande `cd`. Une fois dans le répertoire du projet, activez l'environnement virtuel en exécutant :
   shell

```
source mon_env_python3.10/bin/activate
```

Vous remarquerez que le nom de l'environnement virtuel (`mon_env_python3.10`) apparaît avant le prompt de votre terminal, indiquant que l'environnement est actif.

Lancer le jeu Toujours dans le terminal, lancez le jeu en exécutant la commande suivante :
shell

```
python3.10 main.py --size 10 --games 10 --player qlearning --other random
```

1. Cette commande démarre le jeu avec les paramètres suivants :
    - `--size 10` définit la taille du plateau à 10x10.
    - `--games 10` spécifie que 10 parties seront jouées.
    - `--player qlearning` sélectionne la stratégie Q-Learning pour le joueur principal.
    - `--other random` définit le joueur adverse pour utiliser la stratégie Random.
2. **Observer les résultats** Après le lancement, le jeu va automatiquement jouer 10 parties en utilisant les stratégies spécifiées. Les résultats de chaque partie seront affichés dans le terminal, indiquant lequel des joueurs (White ou Black) a gagné. Il y'aura également une illustration graphique des résultats.

Une fois que vous avez terminé de jouer ou d'effectuer des tests, vous pouvez désactiver l'environnement virtuel en tapant `deactivate` dans le terminal.

Assurez-vous que tous les paquets nécessaires sont installés dans votre environnement virtuel et que votre fichier `main.py` est correctement configuré pour utiliser les options de ligne de commande.

Étudiants : Moustapha Diop, Fall Khadim

Merci pour la lecture